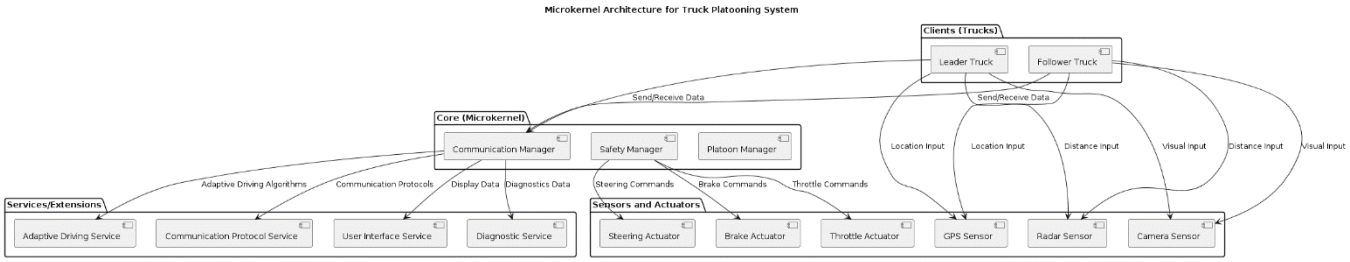
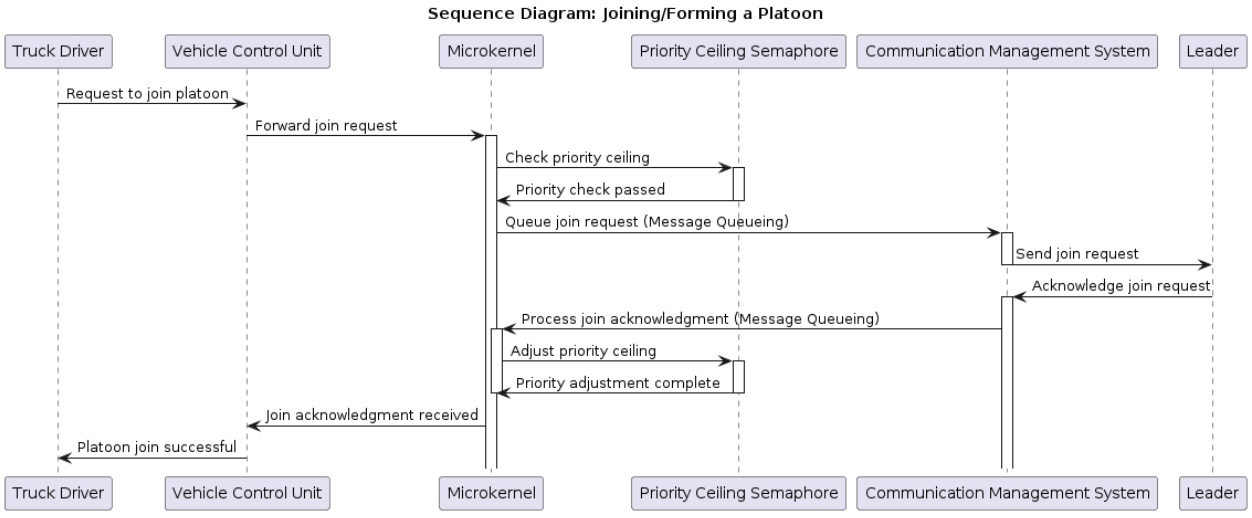
[Only for reference purpose]

Design of a Truck Platooning using Microkernel Architecture Pattern

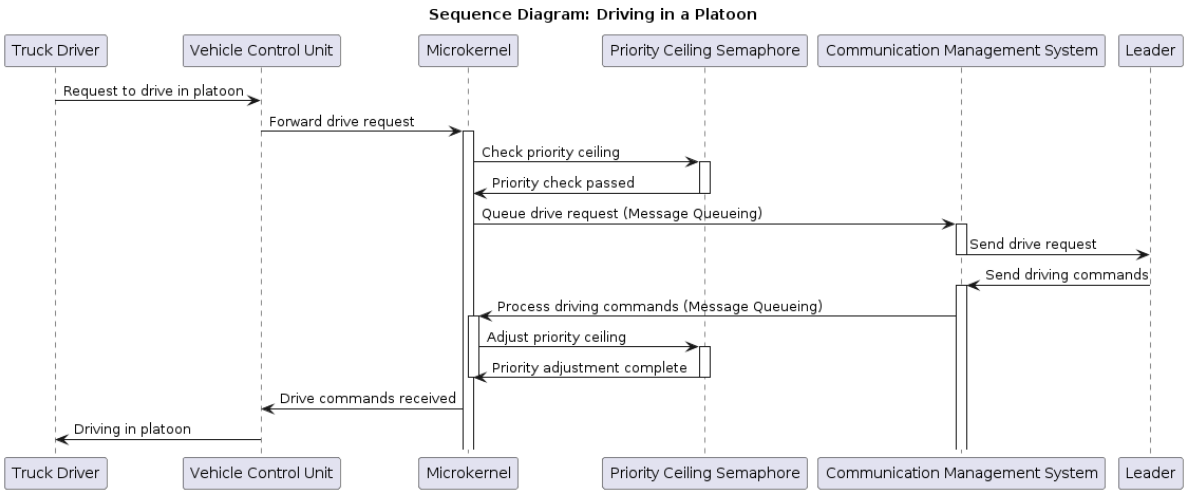
Components Diagram



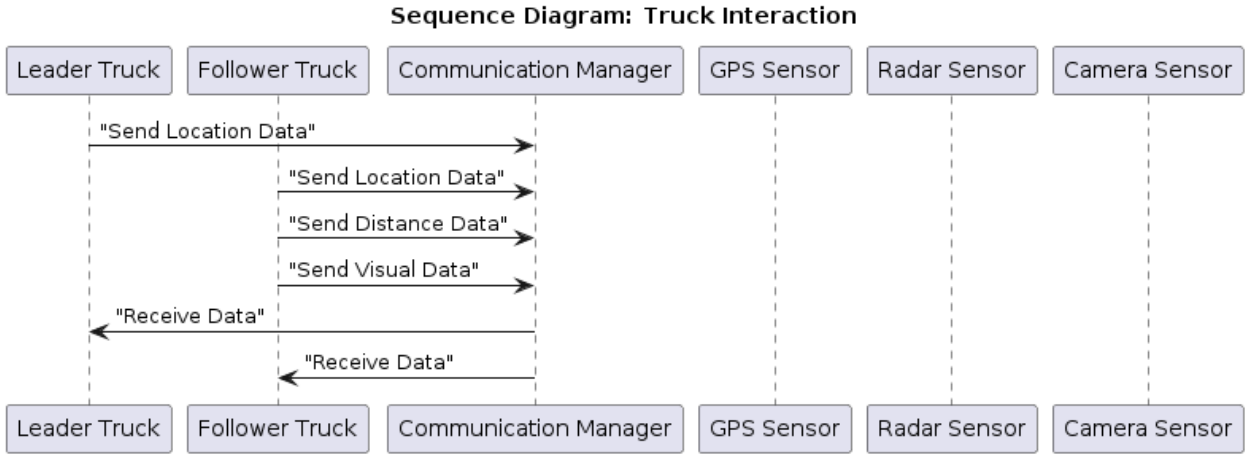
Sequence Diagram Joining-Forming a Platoon



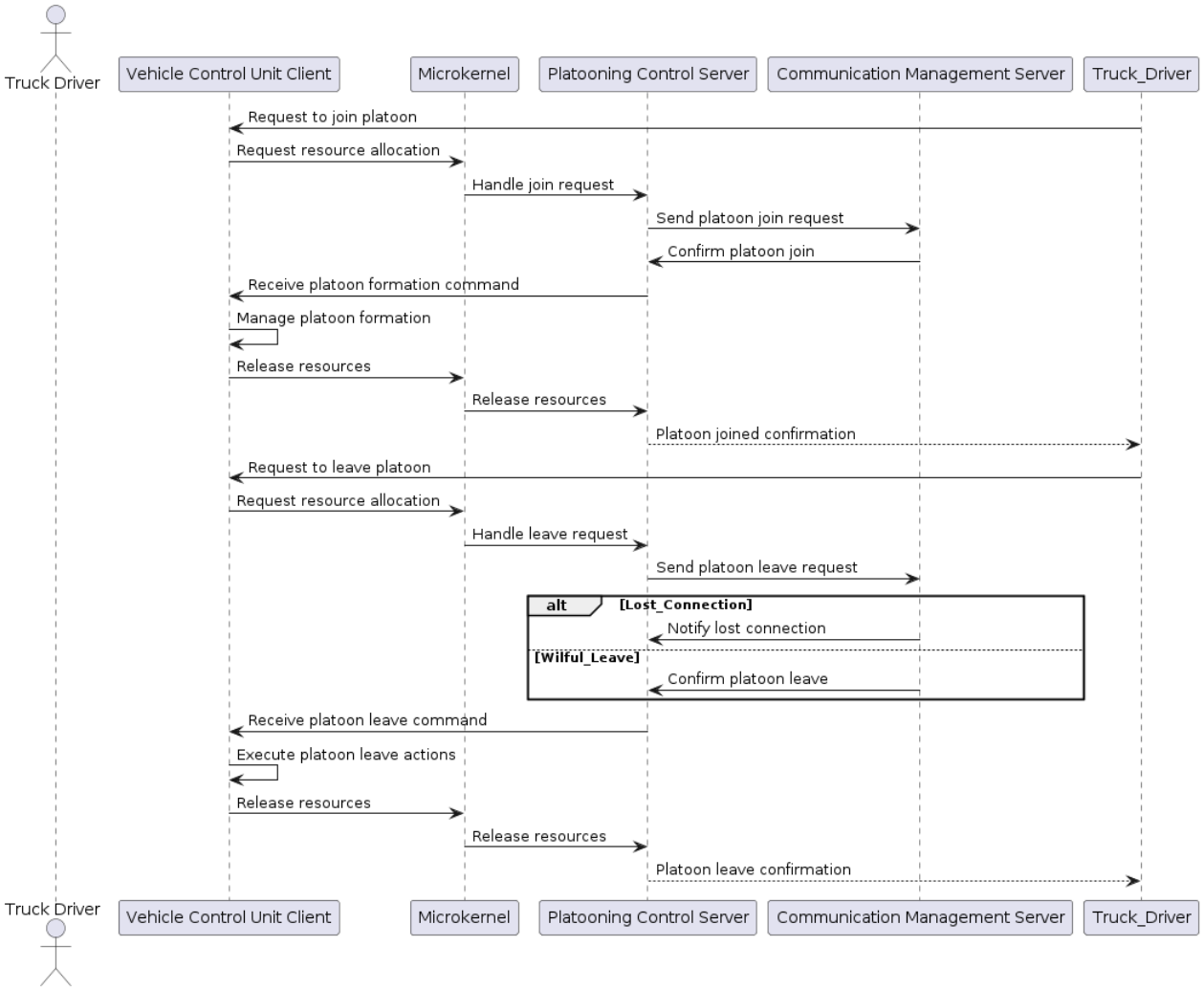
Sequence Diagram Driving in a Platoon with semaphore



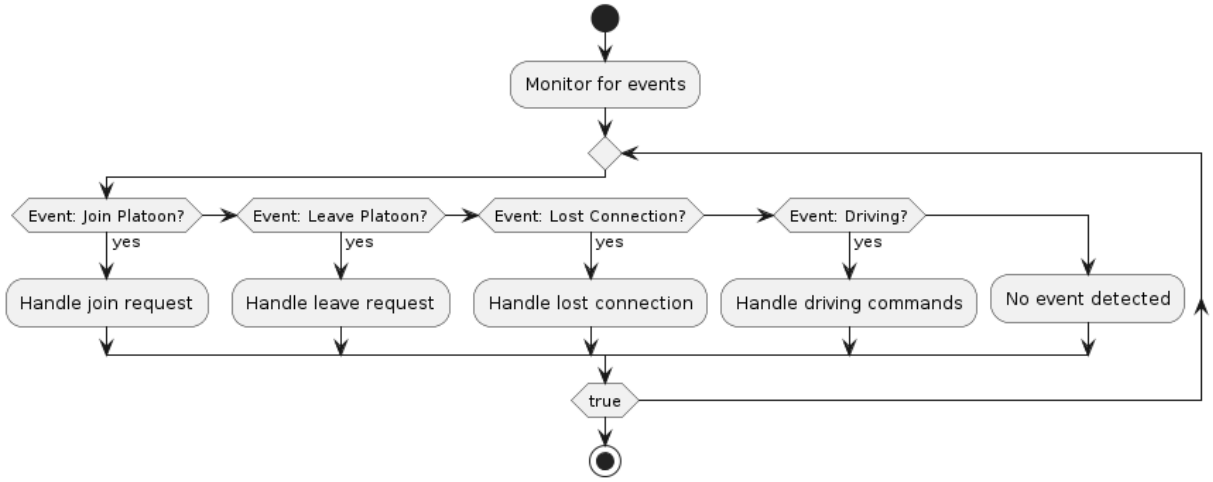
sequence diagram Trucks interactions



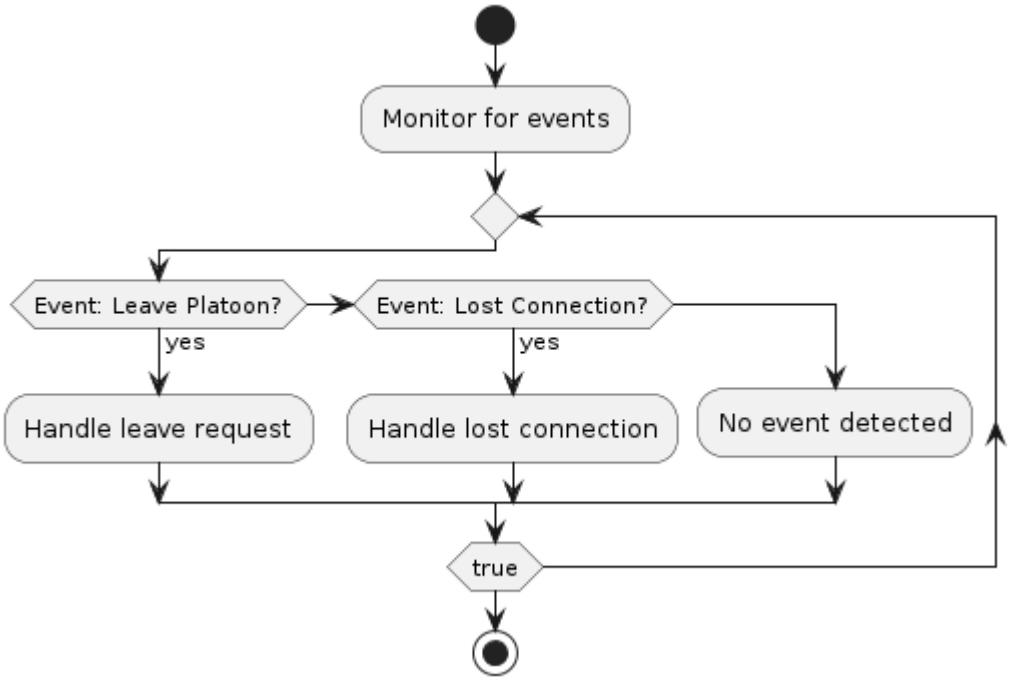
Sequence Diagram Joining or Leaving



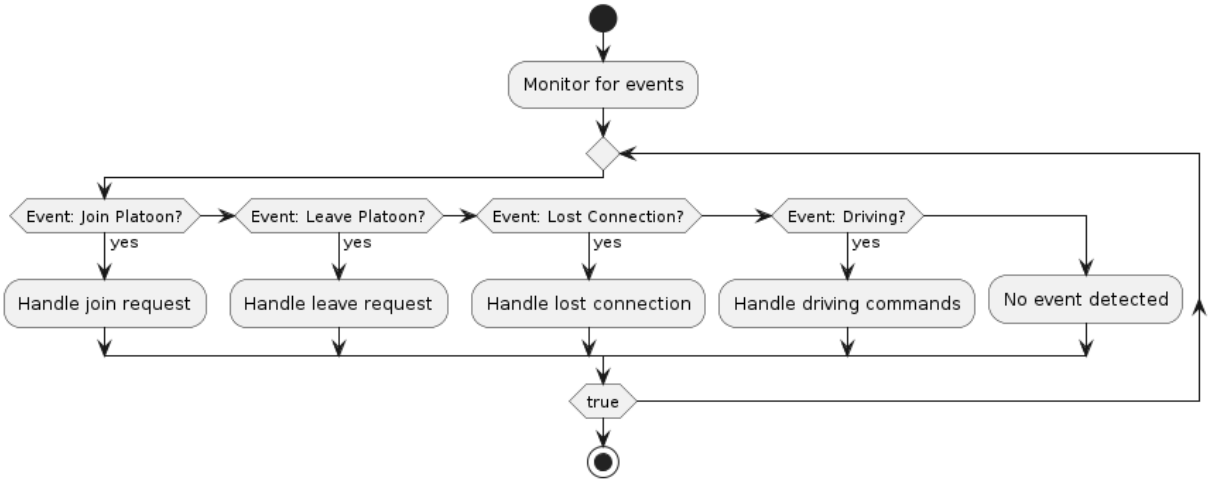
Activity diagram Leader Joining/Forming a Platoon



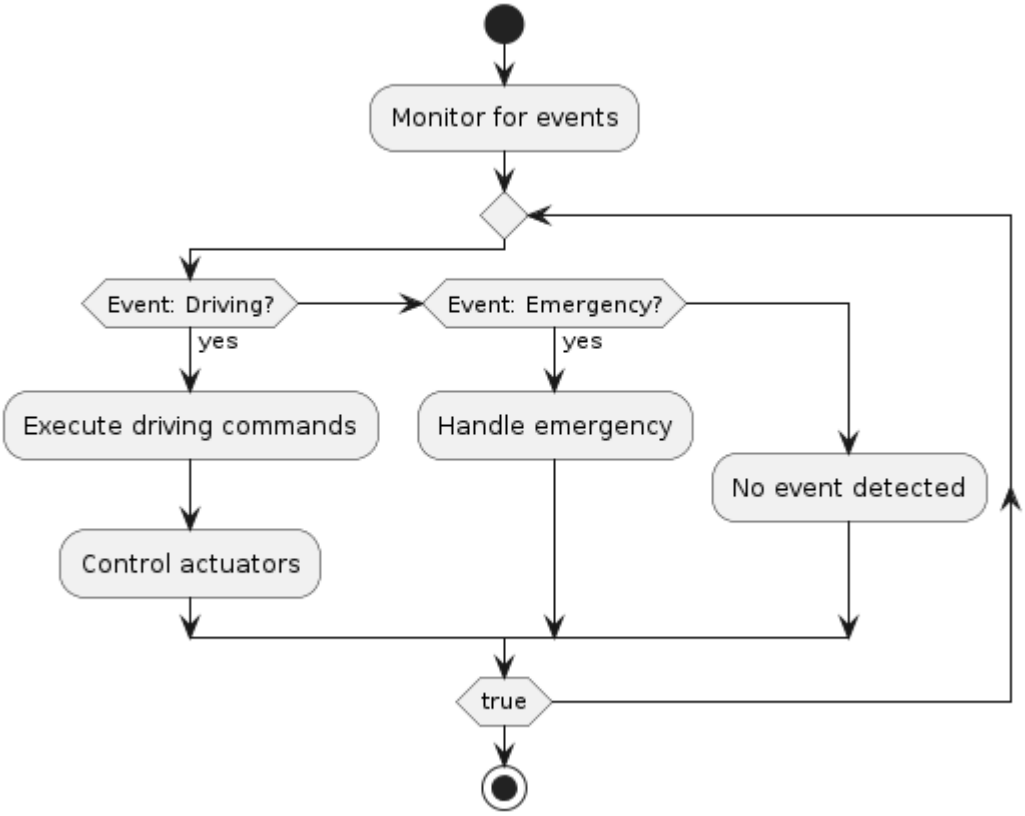
Activity diagram Leader Leaving a Platoon



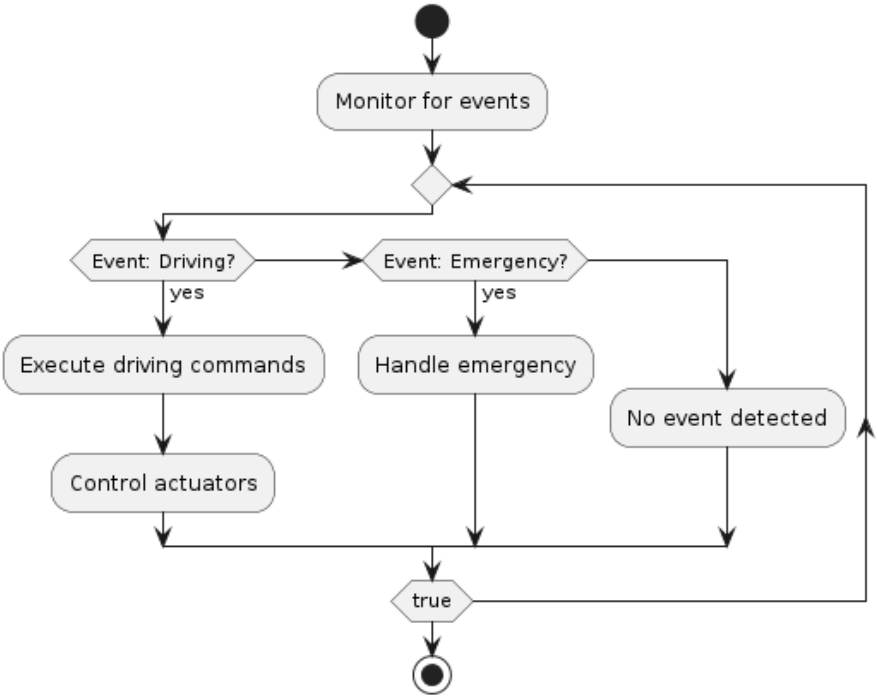
Activity diagram First Follower Joining the Platoon



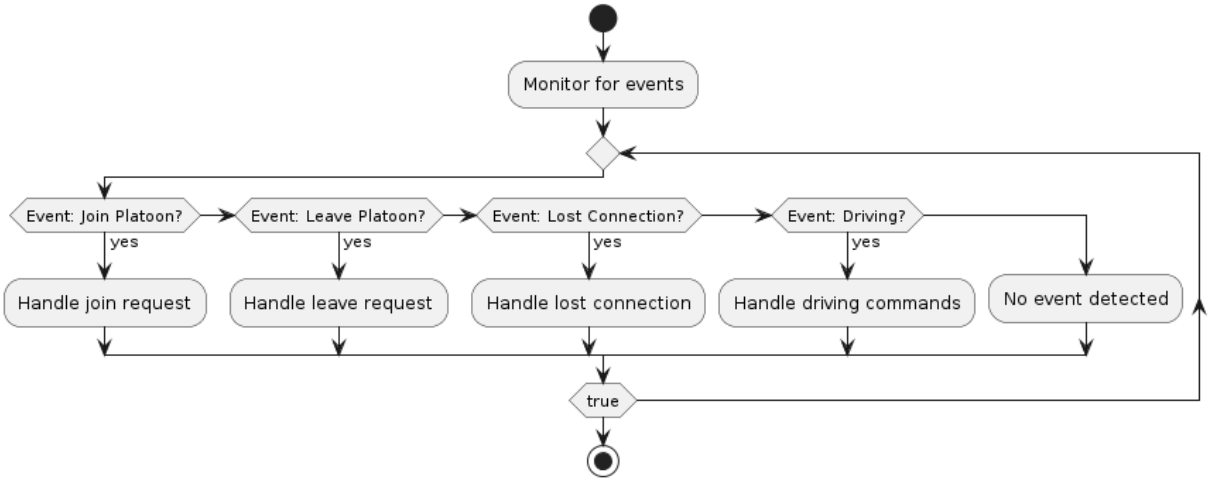
Activity diagram First Follower Driving in a Platoon



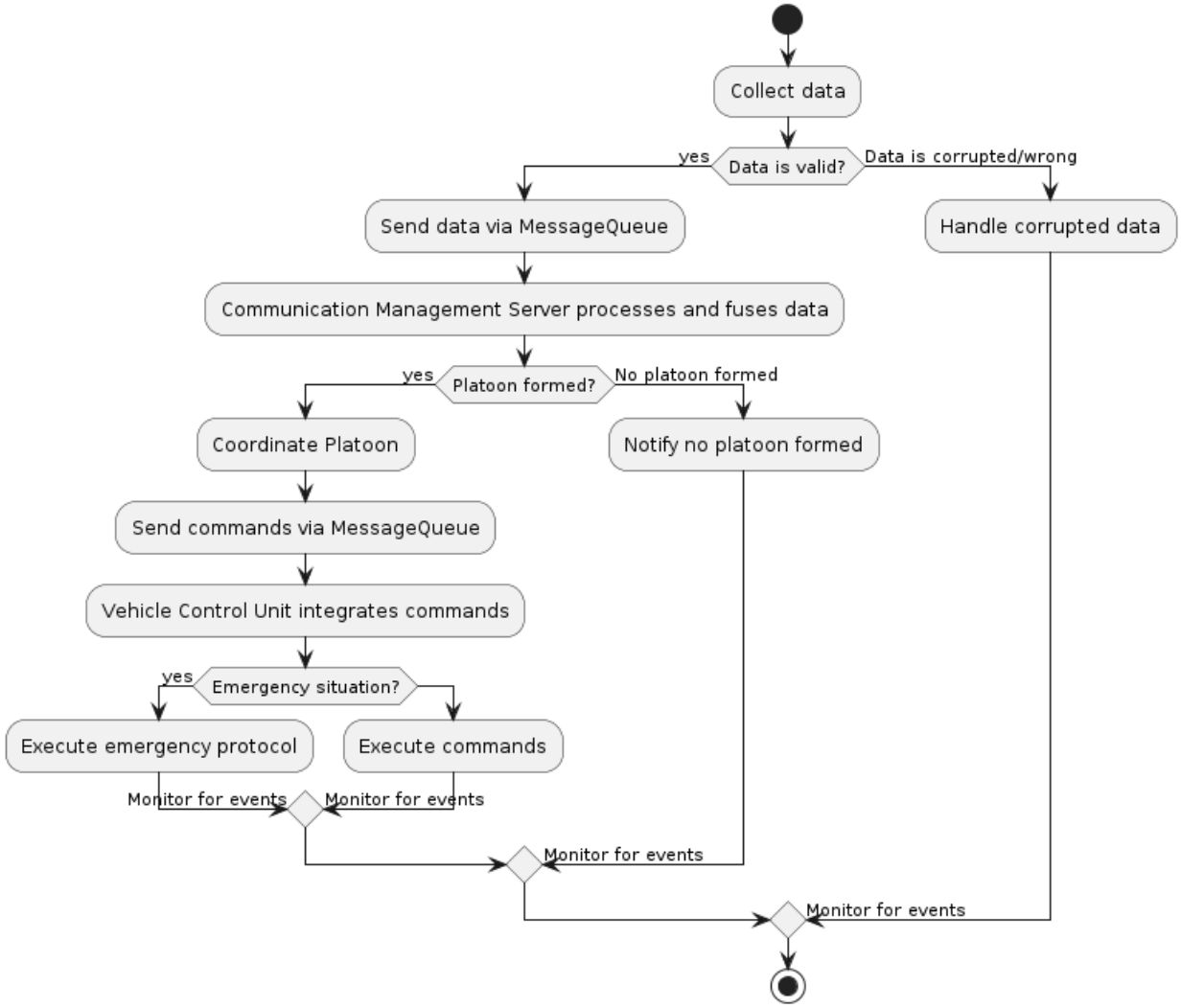
Activity diagram Nth Follower Driving in a Platoon



Activity diagram Nth Follower Joining-Forming a Platoon

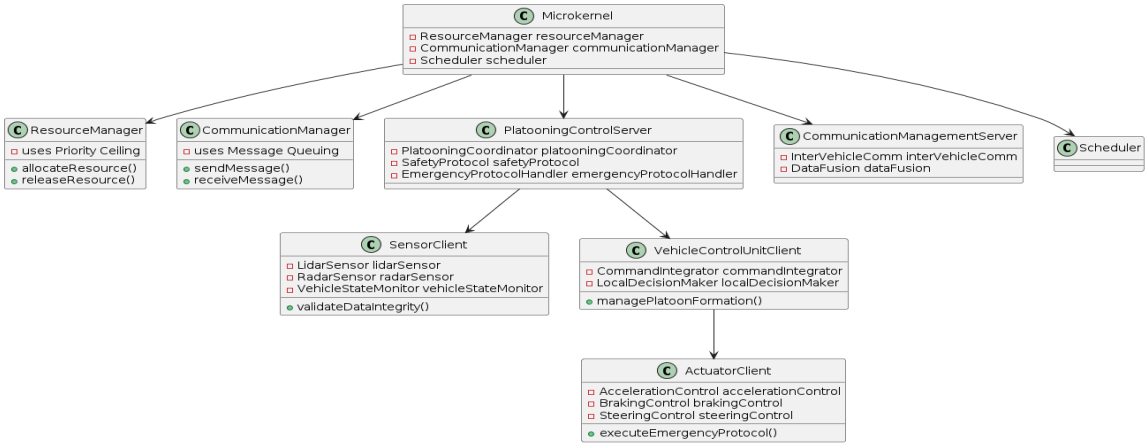


Overall activity Diagram



Class Diagrams

Overall, Class diagram



Class Diagram Leader

Class Diagram first follower

class diagram nth follower

Why the Microkernel Pattern Fits Best

1. Modularity and Extensibility

Definition: The microkernel architecture separates the core system functions (the microkernel) from the system-specific extensions and services.

Application to Platooning: Truck platooning systems require a robust core for essential functions like vehicle communication, safety protocols, and basic driving mechanics. However, they also need to support various extensions like different communication protocols, additional safety features, and adaptive driving algorithms.

Benefit: This modularity allows new features to be added or updated independently of the core system, making the platooning system adaptable and future-proof.

2. Separation of Concerns

Definition: The core microkernel handles only the most critical functions, while additional services handle specific tasks.

Application to Platooning: Core functions include maintaining the platoon formation, ensuring real-time communication, and executing critical safety maneuvers. Extensions handle non-critical functions like user interfaces, diagnostics, and non-essential driving assistance.

Benefit: This separation ensures that critical operations are always prioritized and remain unaffected by failures or changes in non-critical components.

3. Reliability and Fault Isolation

Definition: Faults in extensions do not affect the core microkernel.

Application to Platooning: In a truck platooning system, it is crucial that failures in non-essential services (like a user interface or a diagnostic tool) do not impact the core driving and safety functions.

Benefit: Increases the overall reliability and safety of the system, as critical operations are insulated from faults in less critical services.

Why Other Alternatives Do Not Fit as Well

1. Layered Pattern

Reason: While the layered pattern promotes separation of concerns, it lacks the flexibility and extensibility needed for a highly adaptive system like truck platooning. Changes in one layer can impact others, making it less suitable for dynamic updates and extensions.

2. Component-Based Architecture

Reason: Although it offers modularity, it doesn't inherently provide the strong separation between core and non-core functions that the microkernel architecture does. This can lead to increased complexity in managing critical operations and extensions.

3. Hierarchical Control Pattern

Reason: This pattern focuses on control hierarchies, which may not offer the same level of flexibility and fault isolation as the microkernel pattern. It can be less adaptable to the changing needs and dynamic nature of platooning systems.

Mapping Roles of the Architectural Design Pattern to the Modeled Example

Microkernel: Manages core functions like inter-vehicle communication, platoon formation, and safety protocols.

Server/Services: Handle additional functions such as user interfaces, diagnostics, adaptive driving algorithms, and communication protocol variations.

Client: Trucks (both leader and followers) that interact with the core microkernel and services for platoon management and driving operations.

Describe the Functionality of Each Component

1. Microkernel (Core)

Functionality: Ensures real-time communication between trucks, manages the formation and maintenance of the platoon, and executes critical safety protocols. It provides the fundamental infrastructure upon which other services operate.

Components:

Communication Manager: Manages inter-vehicle communication and data exchange.

Safety Manager: Ensures compliance with safety protocols and handles emergency maneuvers.

Platoon Manager: Maintains platoon structure, including leader and follower roles.

2. Services/Extensions

Functionality: Provide additional, non-critical functionalities that enhance the platooning system. These can be independently developed and updated without impacting the core system.

Examples:

User Interface Service: Allows drivers to interact with the platooning system.

Diagnostic Service: Monitors system health and performance.

Adaptive Driving Service: Implements advanced driving algorithms to optimize fuel efficiency and safety.

Communication Protocol Service: Supports different communication standards and protocols.

3. Client (Trucks)

Functionality: Act as consumers of the services provided by the microkernel and its extensions. Each truck in the platoon (leader and followers) interacts with the core system to receive instructions, communicate status, and perform driving tasks.